Technical Note: Fast Random Projections and Locality Sensitive Hashing via HD

1. Setup

Let:

- H be the normalized Walsh–Hadamard transform
- D = diag(±1) with random independent signs

Define the mapping:

$x \mapsto y = HDx$

This serves as a fast, structured alternative to dense random projections.

---

2. Geometric interpretation

- H is a reflection (eigenvalues ±1)
- D is also a reflection (sign flips along coordinate axes)

So:

> the composition HD is a rotation (more precisely, a product of reflections decomposing into rotations in invariant 2D planes)

Thus:

- HD acts as a **structured random rotation** in high dimensions

---

3. Connection to random projections

Classical random projection:

$$y = Gx, \quad G_{ij} \sim \mathcal{N}(0,1)$$

Structured alternative:

$$y = HDx$$

Properties:

- orthogonal transform → norm preserved
- fast computation O(nlogn)
- no need to store dense matrix

This is closely related to the **Fast Johnson–Lindenstrauss Transform (FJLT)**.

---

4. Distributional behavior

Each output coordinate:

$$y_i = \frac{1}{\sqrt{n}} \sum_j \pm x_j$$

So:

- sum of many signed contributions
- approximately Gaussian when:
    - dimension is sufficiently large
    - input is not too sparse

This follows from central limit behavior.

---

5. Multiple HD layers

Consider:
$y = HD_2HD_1x$

Then:

- composition of rotations
- equivalent to a single orthogonal transform

Key observation

- entries of the equivalent matrix:
  - become sums of ±1 contributions
  - follow a binomial-like distribution
  - converge toward Gaussian as depth increases

So:

> stacking HD layers increases similarity to dense Gaussian projections

---

6. Practical trade-offs

- 1 layer:
  - very fast
  - already useful
- 2 layers:
  - significantly better mixing (often ideal)
- many layers:
  - closer to Gaussian
  - diminishing returns
  - increased cost

---

7. Sparse input behavior

For sparse input (e.g. $x = e_i$):

- HDx produces a Walsh pattern
- all output magnitudes equal

Implication:

- poor discrimination in a single layer

Mitigation:

- apply additional HD layer(s):

  HD_2HD_1x

This:

- breaks uniformity
- improves randomness

---

8. Dimensionality considerations

- higher dimension improves Gaussian approximation
- central limit effects require sufficient mixing width
- very low dimension → poor approximation

---

9. Locality Sensitive Hashing (LSH)

Standard cosine LSH:

$$h(x) = \text{sign}(g^\top x), \quad g \sim \mathcal{N}(0, I)$$

Structured version:

$$y = HDx, \quad h(x) = \text{sign}(y)$$

So:

LSH is obtained by binarizing a random projection

---

10. Angular similarity preservation

For two vectors x, y:

- projection approximately preserves angles

- sign hashing satisfies:

$$\Pr[h(x) = h(y)] \approx 1 - \frac{\theta}{\pi}$$

Thus:

- Hamming distance ≈ angular distance

---

11. Binary vs continuous projection

- continuous projection:
  - retains magnitude information
- binary projection:
  - keeps only sign
  - introduces fixed distortion
  - but preserves similarity statistically

In high dimension:

- binary vector has a predictable relationship to the original projection direction

---

12. Practical guidelines

- use 1–2 HD layers in most cases
- increase depth for very sparse inputs
- ensure sufficient dimensionality
- normalize inputs when using angular similarity

---

13. Applications

- approximate nearest neighbor search
- large-scale similarity search
- dimensionality reduction
- binary embeddings and hashing
- sketching and streaming algorithms
- neural network random layers

- compressed sensing (structured sensing matrices)

---

14. Conceptual summary

- HD = fast structured random rotation
- stacking HD → increasingly Gaussian-like transforms
- binarization → locality sensitive hashing

---

15. Key insight

> Random projection can be implemented as a sequence of simple reflections whose composition approximates a random rotation; binarizing that rotation yields locality-sensitive hashing.